ARM® Performance Monitoring Architecture version 2 Virtualization Extensions

ARM Architecture Group

Document number: Derived from: DSA09-PRDC-010447 6.0
Date of Issue: 22 October, 2010
Author: ARM Limited

In this document, where the term ARM is used to refer to the company it means “ARM or any of its subsidiaries as appropriate”.

Note

The term ARM is also used to refer to versions of the ARM architecture, for example ARMv6 refers to version 6 of the ARM architecture. The context makes it clear when the term is used in this way.

Contents

1 ABOUT THIS DOCUMENT

The material in this document will be incorporated in the next release of the ARMv7-AR Architecture Reference manual (Revision C).It is at a Beta specification state. Please communicate any errata or issues to errata@arm.com.

1.1 Change history

Virtualization Extension specific extracts from DSA09-PRDC-010447 6.0. Other subject matter from this document is available as ARM *Architecture Reference Manual, Performance Monitors v2 Supplement*.

1.2 References

This document refers to the following documents.

Reference	Author(s)	Title
ARM DDI 0406	ARM Limited	ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition
PRD03-GENC-008353	ARM Limited	Virtualization Extensions Architecture Specification
PRD03-GENC-008469	ARM Limited	Large Physical Address Extensions Specification
ARM DDI 0457	ARM Limited	ARM Architecture Reference Manual, Performance Monitors v2 Supplement
events/arm/armv7/events	Jean Pihet	ARMv7 event list for Oprofile

2 INTRODUCTION

2.1 Goals

Performance monitors are *essential* in high-performance, multi-core SoCs. Complex, multi-core, SoCs and computer systems are becoming increasingly hard to model; whether “on the back of an envelope” or in software. These systems comprise multiple interacting and reacting components.

Extensions to the ARMv7 performance monitors aim to improve the capabilities of all ARM architecture processors.

The two main areas for improvement over ARMv7 are:

- allow better directed profiling by an operating system based software monitor.
- provide more uniformity between implementations by extending the list of standard event types to include more common, useful events.

The general additions are documented in the *ARM Architecture Reference Manual, Performance Monitors v2 Supplement*.

2.2 Terminology

This document uses the following terms to describe modes:

- In Secure state, all modes other than User mode are referred to as *Secure privileged modes.*

  That is, System, Supervisor, Abort, Undefined, FIQ, IRQ and Monitor modes.
- In Non-secure state, all modes other than User mode and Hyp mode (if Virtualization Extensions are implemented) are referred to as *Non-secure kernel modes.*

  That is, System, Supervisor, Abort, Undefined, FIQ and IRQ modes.
- *Hyp mode* is only accessible in Non-secure state.

2.3 Support for the Virtualization Extensions

These changes are in addition to the extensions documented in the *Virtualization Extensions Architecture Specification*.

The extensions to profiling to support virtualization form part of PMUv2. The virtualization-specific aspects are described in detail in this document, and are only implemented on processors that implement the ARMv7-A Virtualization Extensions.

Implementations of the ARMv7-A Virtualization Extensions are required to implement PMUv2.

In order that a hypervisor can reserve some counters for closed-loop machine control without having to co-operate with its Guest OSes and still present the illusion of a virtual PMU conforming to the PMU architecture definition to those Guest OSes, the Virtualization Extensions includes a mechanism for reducing the number of counters available to a Guest OS.

3 PMU VIRTUALIZATION EXTENSIONS

3.1 Performance monitor virtualization

Virtualization of the PMU comprises three parts:

1. The hypervisor can trap any access to the PMU by the guest OS. This allows lazy switching of PMU state by a hypervisor. Full lazy switching is supported, but the overhead involved would normally mean this approach is not recommended. However, this mechanism also allows the hypervisor to identify PMU clients and employ intelligent switching of the PMU state.
2. The hypervisor can trap accesses to PMCR, allowing it to fully virtualize the PMU identity registers, PMCR.IMP and PMCR.IDCODE.
3. The hypervisor can reserve the highest-numbered counters for its own use by overriding the value of PMCR.N seen by the guest OS. The processor does not allow the guest OS access to the reserved counters.

Virtualization is controlled by the Hypervisor Debug Configuration Register, HDCR.

3.1.1 Hypervisor Debug Configuration Register, HDCR

HDCR is a new register defined by the *Virtualization Extensions Architecture Specification*.

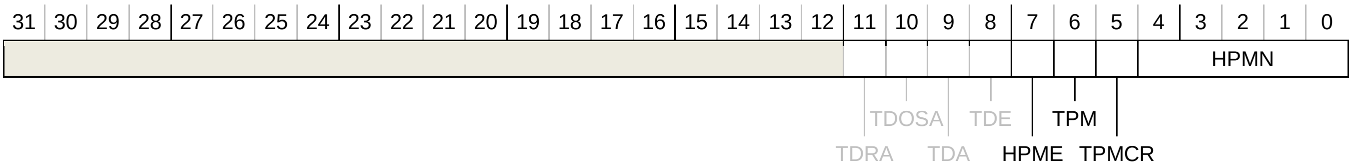

Figure 1: Hypervisor Debug Configuration Register bit assignments

The HDCR defines four fields for PMU register access control, HPME, TPM, TPMCR and HPMN.

HPME, bit [7]

Hypervisor Performance Monitor enable override bit. Overrides the value of PMCR.E for counters configured for access in Hyp mode only. Possible values are:

0b0 Counters in the range [HDCR.HPMN..PMCR.N) are disabled.

0b1 Counters in the range [HDCR.HPMN..PMCR.N) are enabled if also enabled in PMCNTEN.

The reset value of this bit is UNKNOWN.

Note

HDCR.HPME is active in both the Secure and Non-secure state.

TPM, bit [6]

Trap performance monitor accesses. Possible values for this field are:

0b1 Access to performance monitors in Secure privileged modes and Hyp mode only. Accesses from Non-secure kernel and Non-secure User mode cause a Hyp Trap exception.

0b0 Access to performance monitors in all modes.

This bit is ignored in Secure state and in Hyp mode. The reset value of this bit is 0.

TPMCR, bit [5]

Trap performance monitor control register accesses. Possible values for this field are:

0b1 Access to PMCR in Secure privileged modes and Hyp mode only. Accesses from Non-secure kernel and Non-secure User mode cause a Hyp Trap exception.

0b0 Access to PMCR in all modes.

This bit is ignored in Secure state and in Hyp mode. The reset value of this bit is 0.

HPMN, bits [4:0]

Performance Monitor count. Specifies the number of performance counters that can be accessed by the Guest OS.

Reads of PMCR from Non-secure kernel modes return the value of HPMN for PMCR.N.

Possible values are:

0b0000 Reserved.

0b0001 Access to all PMN0 related controls in Non-secure kernel modes. Access to all other performance monitors in Secure and Hyp modes only.

0b0010 Access to all PMN0 and PMN1 related controls in Non-secure kernel modes. Access to all other performance monitors in Secure and Hyp modes only.

0b0011 Access to all PMN0, PMN1 and PMN2 related controls in Non-secure kernel modes. Access to all other performance monitors in Secure and Hyp modes only.

…

PMCR.N

Access to all performance monitor related controls in all modes.

Writing a value greater than the IMPLEMENTATION DEFINED value of PMCR.N is UNPREDICTABLE. The cycle counter, PMCCNTR, is always accessible in Secure privileged modes, Hyp mode and, unless trapped HDCR.TPM == 0b1, Non-secure kernel modes.

Where a Guest OS provides User mode access to the performance monitors through PMUSERENR, the access to the counters from Non-secure User mode is the same as that for Non-secure kernel modes.

Where a Secure OS provides User mode access to the performance monitors through PMUSERENR, the access to the counters from Secure User mode is the same as that for Secure privileged modes.

In Secure state and in Hyp mode, PMCR.N returns the actual number of performance counters.

The reset value of this bit is PMCR.N.

See also *Access permissions* on page 8.

HDCR can only be accessed in Secure privileged modes and Non-secure Hyp mode. Accesses to HDCR from User mode and Non-secure kernel modes are UNDEFINED.

3.1.2 CP15 c9, Performance Monitor Control Register

The extensions interact with fields defined by the performance monitor control register, as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMP								IDCODE								N										DP	X	D	C	P	E

Figure 2: PMCR bit assignment

IMP, IDCODE, DP, X, D, and C, bits [31:16,5:2]

See *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

N, bits [15:11]

Number of event counters. For a definition, see *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

Virtualization Extensions only:

In Non-secure modes other than Hyp mode, this reads the value of HDCR.HPMN. See *Hypervisor Debug Configuration Register, HDCR* on page 6.

In Secure state and Hyp mode, it reads the IMPLEMENTATION DEFINED number of event counters, as defined in the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

P, bit [1]

Event counter reset. For a definition, see *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

Virtualization Extensions only:

In Non-secure modes other than Hyp mode, writing a one to this bit does not reset event counters reserved for Hyp mode by HDCR.HPMN.

In Secure state and Hyp mode, writing a one to this bit resets all event counters.

E, bit [0]

Enable. For a definition, see *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

Virtualization Extensions only:

This bit does not enable/disable counting by event counters reserved for Hyp mode by PMCR.HPMN. Neither does it suppress generation of performance monitor overflow interrupt requests by those counters. See *Counter enables* on page 8 for more details.

3.2 Counter enables

Table 1 shows, for each counter, PMN*x*, whether it is enabled or disabled according to the values of HDCR.HPME, HDCR.HPMN, PMCNTEN and the PMCR.E bit.

Table entries with a grey background do not apply if Virtualization Extensions are not implemented.

HDCR.HPME	PMCR.E	PMCNTEN[1][*x*] == 0	PMCNTEN[*x*] == 1		
			x < HDCR.HPMN	*x* ≥ HDCR.HPMN	
0	0	PMN*x* disabled	PMN*x* disabled	PMN*x* disabled	
0	1	PMN*x* disabled	PMN*x* enabled	PMN*x* disabled	
1	0	PMN*x* disabled	PMN*x* disabled	PMN*x* enabled	***Virtualization Extensions only***
1	1	PMN*x* disabled	PMN*x* enabled	PMN*x* enabled	

Table 1: Event counter enables

Notes

— The effect of HDCR.HPME and HDCR.HPMN on counter enables applies in both the Secure and Non-secure state. However, the value returned for PMCR.N is not affected by HDCR.HPMN in Secure state.

— Generation of overflow interrupt requests occur when the *counter* that has overflowed is enabled according to Table 1, regardless of the value of PMCR.E.

The enables for PMCCNTR are not affected by HDCR, as shown in Table 2.

PMCR.E	PMCNTEN[31] == 0	PMCNTEN[31] == 1
0	PMCCNTR disabled	PMCCNTR disabled
1	PMCCNTR disabled	PMCCNTR enabled

Table 2: Cycle counter enable

If Virtualization Extensions are not implemented, the counter enables for PMN*x* and PMCCNTR are as defined in the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

3.3 Access permissions

Normally the performance monitor registers are accessible from Secure privileged modes, Hyp mode and Non-secure kernel modes.

However, the access permissions for PMU registers can be modified by changing the HDCR and PMUSERENR registers:

- setting the PMUSERENR.EN bit to 1 permits access from User mode
- setting HDCR.TPM to 1 disables access to all registers from Non-secure kernel modes and Non-secure User modes
- setting HDCR.TPMCR to 1 disables access to PMCR from Non-secure kernel modes and Non-secure User modes.

The performance monitor registers are in 4 groups, in no particular order. See *PMUv2 Register Map* on page 12. (A final group represents read accesses to two write-only registers. These operations are reserved, but included here for completeness.)

For each register for which the *access permission* is "Proceed" in the current mode and state, the behavior on reads or writes further depends on what *counter access* is granted for each counter in the current mode and state.

[1] PMCNTEN is the underlying register that writes to PMCNTENSET and PMCNTENCLR set/clear bits in. Reads of PMCNTENSET and PMCNTENCLR return the PMCNTEN value.

In Table 4 through to Table 8, "UNDEFINED" means an Undefined Instruction exception is generated that is taken locally, and "Hyp Trap" means a Hyp Trap Exception is generated with an Undefined Instruction syndrome. "||" means the result of the logical OR function on the two values.

Notes

- Table entries with a grey background do not apply if Virtualization Extensions are not implemented.
- If Virtualization Extensions are not implemented, these access permissions are unchanged from those described in revision B of the *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

Access from Secure privileged modes and Hyp mode

All defined instructions are always accessible in Secure privileged modes and Hyp mode.

HDCR. TPM	HDCR. TPMCR	PMUSERENR. EN	Secure privileged modes	Non-secure Hyp mode	
X	X	X	Proceed	Proceed	***Virtualization Extensions only***

Table 3: Access permissions for all registers in Secure privileged modes and Hyp mode

Group 1

Table 4 summarizes the access permissions for registers in group 1, that is:

- PMUSERENR, `MSR` only
- PMINTENSET, `MRS` and `MSR`
- PMINTENCLR, `MRS` and `MSR`

These registers are never accessible in User mode. Accesses from Non-secure kernel modes are trapped to the hypervisor when HDCR.TPM is set.

HDCR.		PMUSERENR. EN	Secure User mode	Non-secure modes		
TPM	TPMCR			Kernel	User	
0	X	X	UNDEFINED	Proceed	UNDEFINED	
1	X	X	UNDEFINED	Hyp Trap	UNDEFINED	***Virtualization Extensions only***

Table 4: Access permissions for Performance Monitor registers, group 1

Group 2

Table 5 summarizes the access permissions for registers in group 2, that is

- PMUSERENR, `MRS` only.

This register is normally readable in User mode. Accesses from Non-secure modes are trapped to the hypervisor when HDCR.TPM is set.

HDCR.		PMUSERENR. EN	Secure User mode	Non-secure modes		
TPM	TPMCR			Kernel	User	
0	X	X	Proceed	Proceed	Proceed	
1	X	X	Proceed	Hyp Trap	Hyp Trap	***Virtualization Extensions only***

Table 5: Access permissions for Performance Monitor registers, group 2

Group 3

Table 6 summarizes the access permissions for registers in group 3, that is:

- PMCR, `MSR` and `MRS` only.

PMCR is normally read/write in User mode only when User mode access is enabled, otherwise reads of and writes to PMCR are locally UNDEFINED. Accesses to PMCR from Non-secure modes that are not locally UNDEFINED are trapped to the hypervisor whenever either HDCR.TPM or HDCR.TPMCR is set.

HDCR.		PMUSERENR. EN	Secure User mode	Non-secure modes	
TPM	TPMCR			Kernel	User
0	0	0	UNDEFINED	Proceed	UNDEFINED
0	0	1	Proceed	Proceed	Proceed

HDCR. TPM	HDCR. TPMCR	PMUSERENR. EN	Secure User mode	Non-secure modes Kernel	Non-secure modes User	
0	1	0	UNDEFINED	Hyp Trap	UNDEFINED	*Virtualization Extensions only*
1	X	0	UNDEFINED	Hyp Trap	UNDEFINED	
0	1	1	Proceed	Hyp Trap	Hyp Trap	
1	X	1	Proceed	Hyp Trap	Hyp Trap	

Table 6: Access permissions for Performance Monitor registers, group 3

Group 4

Table 7 summarizes the access permissions for registers in group 4, that is:

- PMCNTENSET, MRS and MSR
- PMCNTENCLR, MRS and MSR
- PMOVSR, MRS and MSR
- PMSELR, MRS and MSR
- PMCCNTR, MRS and MSR
- PMXEVTYPER, MRS and MSR
- PMCCFILTR, MRS and MSR
- PMXEVCNTR, MRS and MSR
- PMSWINC, MSR only
- PMCEID0, MRS only, version 2 only
- PMCEID1, MRS only, version 2 only.

These registers are normally accessible in User mode when User mode access is enabled, otherwise accesses are locally UNDEFINED. Accesses to these registers from Non-secure modes that are not locally UNDEFINED are trapped to the hypervisor when HDCR.TPM is set.

HDCR. TPM	HDCR. TPMCR	PMUSERENR. EN	Secure User mode	Non-secure modes Kernel	Non-secure modes User	
0	X	0	UNDEFINED	Proceed	UNDEFINED	
0	X	1	Proceed	Proceed	Proceed	
1	X	0	UNDEFINED	Hyp Trap	UNDEFINED	*Virtualization Extensions only*
1	X	1	Proceed	Hyp Trap	Hyp Trap	

Table 7: Access permissions for Performance Monitor registers, group 4

For performance monitors version 1, PMCEID0 and PMCEID1 are reserved registers.

Note

There is no difference between groups 3 and 4 when Virtualization Extensions are not implemented.

Reserved registers

Table 8 determines the access permissions for read access to write-only registers and write access to read-only registers, that is:

- All reserved encodings
- PMSWINC, MRS only
- PMCEID0, MSR only, version 2 only
- PMCEID1, MSR only, version 2 only.

These accesses are reserved.

PMUSERENR.EN	Secure privileged modes, Non-secure kernel modes and Hyp mode	User mode
0	UNPREDICTABLE	UNDEFINED
1	UNPREDICTABLE	UNPREDICTABLE

Table 8: Access permissions for reserved accesses to Performance Monitor registers

3.4 Counter access

Counters are always accessible in Secure privileged modes and Hyp modes. Otherwise if the counter is reserved by the hypervisor using HDCR.HPMN, it cannot be accessed from Non-secure kernel modes and Non-secure User mode.

Note

Even though this section may describe a *counter* as having "Access" from a particular mode and state, access to the *registers* are subject to the access permissions described in *Access permissions* on page 8. In particular, accesses from User mode may be UNDEFINED and accesses from Non-secure kernel and Non-secure User modes may cause a Hyp Trap exception.

3.4.1 PMN*x* event counters

Table 9 shows how access to a counter, PMN*x*, is governed in each mode according to the setting of HDCR.HPMN.

x < HDCR.HPMN	Secure modes		Non-secure modes			
	Privileged	**User**	**Hyp**	**Kernel**	**User**	
Yes	Access	Access	Access	Access	Access	
No	Access	Access	Access	No access	No access	***Virtualization Extensions only***

Table 9: Determining PMN*x* counter access from HDCR

If there is *no access* to a particular counter, PMN*x*, for the current mode and state, then:

- if PMSELR.SEL == *x* then
  - — reads of PMXEVTYPER and PMXEVCNTR return UNKNOWN[2]
  - — writes to PMXEVTYPER and PMXEVCNTR are UNPREDICTABLE[3]
- PMOVSR[*x*], PMCNTENSET[*x*] and PMCNTENCLR[*x*], PMINTENSET[*x*] and PMINTENCLR[*x*] are RAZ/WI
- writes to PMSWINC[*x*] are ignored
- writing 1 to PMCR.P does not reset PMN*x*.

3.4.2 CCNT cycle counter

There is no control to allow the hypervisor to "reserve" the cycle counter for its own use. This would break the virtualization of the performance monitors to the Guest OS.

HDCR.HPMN	Secure modes		Non-secure modes		
	Privileged	**User**	**Hyp**	**Kernel**	**User**
X	Access	Access	Access	Access	Access
					Virtualization Extensions only

Table 10: Determining cycle counter access

However, accesses to the PMCCNTR register are subject to the access permissions described in *Access permissions* on page 8. In particular, accesses from User mode may be UNDEFINED and accesses from Non-secure kernel or User modes may cause a Hyp Trap exception.

[2] The definition of UNKNOWN means that this must not be a security or privilege hole. Note that returning the actual value of the counter does not usually constitute a security or privilege hole.

[3] The definition of UNPREDICTABLE means this must not be a security or privilege hole. In particular, the write must not change the value in the register.

4 PMUV2 REGISTER MAP

4.1 CP15 c9

Table 11 gives the decode table for CP15 `MRC` and `MCR` instructions with `opc1==0`, `CRn==c12`, `c13` or `c14`, and `CRn==c9`, that is, instructions of the form `MRC|MCR p15,0,c9,<CRm>,<opc2>`.

CRm	opc2	Access	Grp	Vsns	Description
c12	0	`MRC` or `MCR`	3	1,2	*CP15 c9, Performance Monitor Control Register* on page 7
	1	`MRC` or `MCR`	4	1,2	PMCNTENSET, Count Enable Set Register
	2	`MRC` or `MCR`	4	1,2	PMCNTENCLR, Count Enable Clear Register
	3	`MRC` or `MCR`	4	1,2	PMOVSR, Overflow Flag Status Register
	4	`MCR`	4	1,2	PMSWINC, Software Increment Register
		`MRC`	-	-	Reserved
	5	`MRC` or `MCR`	4	1,2	PMSELR, Event Counter Selection Register
	6,7	`MRC`	-	1	Reserved
	6	`MRC`	4	2	PMCEID0 Common Event Identification Registers
	7	`MRC`	4	2	PMCEID1
	6,7	`MCR`	-	-	Reserved
c13	0	`MRC` or `MCR`	4	1,2	PMCCNTR, Cycle Count Register
	1	`MRC` or `MCR`	4	1,2	*CP15 c9, Event Type Select Register, PMXEVTYPER* on page 13
				2	*CP15 c9, Cycle Count Filter Control Register, PMCCFILTR* on page 14
	2	`MRC` or `MCR`	4	1	PMXEVCNTR, Event Count Register
	3-7	`MRC` or `MCR`	-	-	Reserved
c14	0	`MCR`	1	1,2	PMUSERENR, User Enable Register
		`MRC`	2		
	1	`MRC` or `MCR`	1	1,2	PMINTENSET, Interrupt Enable Set Register
	2	`MRC` or `MCR`	1	1,2	PMINTENCLR, Interrupt Enable Clear Register
	3-7	`MRC` or `MCR`	-	-	Reserved

Table 11: CP15 c9 register map

“Grp” is the access permission group; see *Access permissions* on page 8. “Vsns” are the PMU architecture versions. Registers whose names are not in italics and are not cross-referenced are not changed by this specification.

4.2 Memory-mapped and external debug interfaces

The *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition* suggests that memory-mapped interface and external debug interfaces to the PMU can be provided, but does not define nor require such interfaces. For details, see *ARM Architecture Reference Manual, Performance Monitors v2 Supplement*.

5 PMUV2 EVENT FILTERING

5.1 Description of requirements

Event filtering allows events to be counted only in certain modes and/or states. Event filtering is described in *ARM Architecture Reference Manual, Performance Monitors v2 Supplement*. The Virtualization Extensions extend event filtering to encompass Hyp mode.

Hyp is a special case because Hyp is *always* used for a layer of specialized control software, and never used for generic applications.

5.2 Updated register definitions

5.2.1 CP15 c9, Event Type Select Register, PMXEVTYPER

PMXEVTYPER is defined as follows:

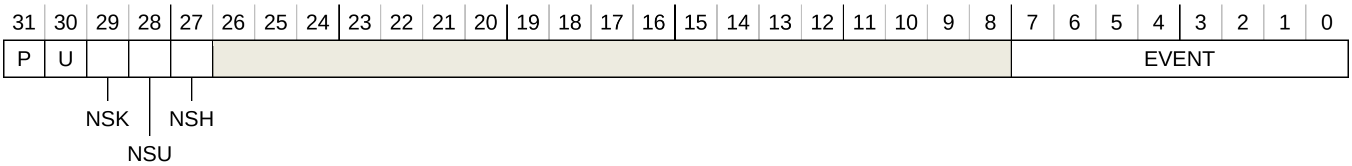

Figure 3: Event Type Select Register bit assignments

Bits [26:8]

Reserved.

P, U, NSK, NSU, bits [31:28]

As defined in *ARM Architecture Reference Manual, Performance Monitors v2 Supplement*.

NSH, bit [27], Virtualization Extensions implemented

Count in Hyp mode. Possible values of this field are:

0 Do not count in Hyp mode.

1 Count in Hyp mode.

The reset value of this bit is UNKNOWN.

Bit [27], Virtualization Extensions not implemented

Reserved.

EVENT, bits [7:0]

As defined (as “evtCount”) in *ARM® Architecture Reference Manual ARMv7-A and ARMv7-R edition*.

Note

Some combinations of P, U, NSK, NSU and NSH are deprecated and must not be selected by software. See *Reserved encodings* on page 13.

Reserved encodings

This encoding supports all possible combinations, including the less useful ones shown in Table 12. The seven cases shown in Table 12 fall into one of three types:

- Never
- User mode in one security state and privileged or kernel modes in the other security state (with or without Hyp mode)
- User mode and Hyp mode without kernel modes.

These combinations are reserved. Software must not select any of these cases. However, they are not UNPREDICTABLE encodings and hardware must implement the five filtering bits as described.

P	U	NSK	NSU	NSH	Modes in which events are counted
1	1	0	0	0	Never
0	1	1	1	0	Secure privileged modes and Non-secure User mode

P	U	NSK	NSU	NSH	Modes in which events are counted
1	0	1	1	0	Secure User mode and Non-secure kernel modes
0	1	1	1	1	Secure privileged modes, Non-secure User mode and Hyp mode
1	0	1	1	1	Secure User mode, Non-secure kernel modes and Hyp mode
1	0	0	0	1	User mode, plus Hyp mode
1	0	0	1	1	Secure User mode, plus Hyp mode
1	1	0	1	1	Non-secure User mode, plus Hyp mode

Table 12: Reserved encodings, do not use

5.2.2 CP15 c9, Cycle Count Filter Control Register, PMCCFILTR

The Cycle Count Filter Control Register, PMCCFILTR configures which modes and states the cycle counter, CCNT, counts in. The format of PMCCFILTR is shown in Figure 4.

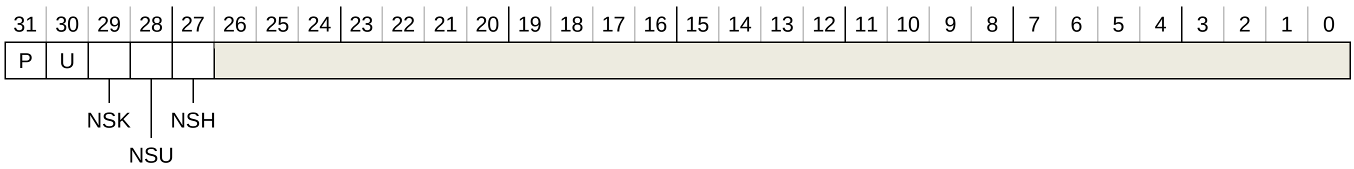

Figure 4: Cycle Count Filter Control bit assignments

Bits [26:0]

Reserved.

P, U, NSK, NSU and NSH, bits [31:27]

See definitions in *CP15 c9, Event Type Select Register, PMXEVTYPER* on page 13. However, for compatibility with PMU architecture version 1, the reset values of these bits are 0.

Note

PMCCFILTR was not defined in PMU architecture version 1, so existing software will not write to it before enabling the cycle counter. Hence these bits are reset to zero in PMCCFILTR.
This differs from PMXEVTYPER which must be initialized to enable an event. Hence the equivalent bits in PMXEVTYPER have no defined reset value.

